Optimisation de la qualité de service par l'utilisation de mémoire cache

Rémy Leone, Paolo Medagliani, Jérémie Leguay

Optimisation de la qualité de service par l'utilisation de mémoire cache

Rémy Léone, Paolo Medagliani, Jérémie Leguay

Thales Communications and Security, 4 avenue des Louvresses, Gennevilliers, France

Nous nous intéresserons dans cet article à l'utilisation de mémoire cache pour améliorer l'utilisation des ressources dans un réseau de capteurs. Plusieurs clients souhaitent consulter la même information au sein d'un réseau contraint en énergie. Afin d'éviter les requêtes redondantes, nous utiliserons un serveur proxy intelligent capable de décider quand transférer la requête au réseau de capteurs ou d'y répondre directement avec des informations mémorisées pour une durée de validité donnée. Nous verrons d'abord comment l'information est mise à la disposition de ce serveur, puis comment les informations venant d'autres couches logiques peuvent contribuer à prendre des décisions de communications et de configuration plus intelligentes, par exemple comment mettre en place une stratégie d'optimisation qui nous permettra d'augmenter la durée de vie du réseau ou la satisfaction utilisateur, exprimée par la fraicheur des informations disponibles. Nous verrons aussi comment maximiser les deux en utilisant des solutions optimales non dominées de Pareto.

Keywords: Réseaux de capteurs, mémoire cache, CoAP, économies d'énergie

1 Introduction

Les réseaux de capteurs sont une des composantes de l'Internet des objets. Cependant ils sont soumis le plus souvent à des contraintes énergétiques, des liens de communications instables et des taux d'erreurs de transmissions importants. Afin de résoudre ces problèmes, des systèmes d'exploitation [DGV04] et des protocoles adaptés tels que X-MAC, 6LowPAN, RPL et CoAP [She12] sont utilisés. Une mémoire cache est également mise en place à l'entrée des réseaux de capteurs pour éviter de trop solliciter les nœuds.

Nous proposons dans ce travail d'étendre le rôle et la gestion de cette mémoire cache pour adapter l'utilisation et la configuration des ressources disponibles en fonction de la demande. Cet article présente en particulier une stratégie pour améliorer la qualité de service d'un réseau de capteurs en adaptant dynamiquement les paramètres de la mémoire cache en fonction d'informations en provenance de différentes couches logiques.

2 Système de cache avancé (concepts et problématiques)

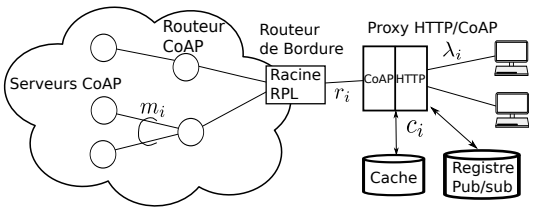

FIGURE 1: Représentation de notre architecture

Comme présenté dans la Figure 1, le réseau est composé d'un ensemble de nœuds connectés entre eux formant un arbre ayant pour racine un routeur de bordure, qui envoie et reçoit les paquets vers l'extérieur. Chaque nœud i a m_i enfants au sein de l'arbre de routage.

La passerelle est un serveur capable de recevoir des requêtes et de les traduire dans un format adapté. Ces requêtes seront ensuite envoyées au routeur de bordure du réseau de capteurs suivant une fréquence r_i que l'on peut mesurer. Nous supposerons par la suite que le rythme des requêtes venant de l'extérieur suit une loi de Poisson de paramètre λ_i.

La passerelle implémente un mécanisme de cache lui permettant de garder en mémoire les résultats des requêtes passées pendant une durée de vie c_i. Une requête concernant une ressource i venant de l'extérieur depuis moins de c_i sera traitée par la passerelle sans solliciter le réseau de capteurs. Étant critiques, ces paramètres c_i doivent être choisis avec précision. Un c_i trop petit rendra la mémoire cache inefficace ; à l'inverse une durée de cache trop longue risquera de donner des valeurs non pertinentes pour les applications.

Stratégiquement placé entre l'offre et la demande, le cache peut contenir d'autres informations utiles telles que la topologie du réseau, les services offerts par les nœuds, les applications extérieures qui les utilisent, les paramètres de la couche MAC, l'énergie résiduelle des nœuds. Dans les approches classiques, ces informations ne sont pas recoupées afin d'agir sur la politique de cache et la configuration du réseau. C'est l'objectif que nous allons poursuivre car des économies d'énergie importantes sont à réaliser quand une vue globale du système est disponible.

3 Performance d'un cache simple

Dans un premier temps, nous étudions le fonctionnement d'un cache simple. Nous pouvons d'ores et déjà voir que les intérêts immédiats du cache comprennent une réduction des délais de réponse et une supervision facilitée de notre réseau de capteur vis a vis des notifications et de la topologie.

3.1 Modèle théorique

Si la ressource demandée par le client n'est pas dans le cache ou que la donnée présente est trop vieille, le serveur va faire une requête sur la ressource en question. Soit T_i le temps moyen entre 2 requêtes arrivant sur la passerelle pour une ressource i, comme énoncé dans [LML13], le temps d'attente entre 2 requêtes consécutives transmises aux capteurs est de :

$$r_i = \begin{cases} \lceil \frac{c_i}{T_i} \rceil T_i & \text{si } T_i \leq c_i \\ T_i & \text{sinon.} \end{cases} \tag{1}$$

Le temps entre chaque requête est modélisé par un temps d'attente exponentiel de paramètre λ_i. Dans la mesure où les requêtes sont aléatoires et uniformément réparties entre les différents nœuds on démontre [LML13] que le temps d'arrivée entre deux requêtes est une loi exponentielle de paramètres $T = NT_i = \frac{N}{\lambda_i}$.

On peut définir MR (cache Miss Ratio) comme la probabilité qu'une requête ne puisse pas être satisfaite par les informations disponibles dans le cache. Ainsi, MR correspond à la probabilité que le temps entre chaque arrivée d'une requête soit plus grande que c_i :

$$MR = \int_{c_i}^{\infty} \frac{e^{-\frac{t}{T}}}{T} \mathrm{d}t = e^{-\frac{c_i}{T}}. \tag{2}$$

Ainsi on déduit CH (Cache Hit) qui est le complémentaire de MR : $CH = 1 - e^{-\frac{c_i}{T}}$.

3.2 Implémentation et résultats de la simulation

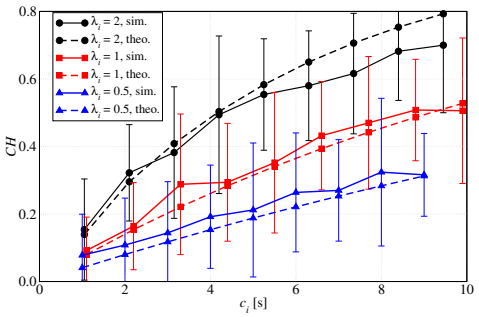

FIGURE 2: Cache Hit (CH) en fonction de c_i, durée de validité d'une information.

Nous considérons que la durée de vie du réseau est *le temps qui sépare son démarrage du premier arrêt définitif d'au moins un des nœuds*, que tous les nœuds démarrent avec la même énergie initiale. Le réseau que nous utilisons est composé de 12 nœuds fixes connectés via un arbre. Des requêtes sont générées et envoyées vers la passerelle qui décide de celles à traiter en utilisant le cache. Les nœuds économisent de l'énergie en utilisant ContikiMAC en tant que protocole d'accès à la couche de liaison. Ce protocole introduit des cycles de veille pour allumer périodiquement l'interface radio. On

suppose que les transmissions sont effectives et sans erreurs. On suppose que la simulation commence lorsque l'arbre de routage RPL a convergé.

Dans la Figure 2 nous pouvons voir que plus λ_i est élevé, plus le cache est intéressant. Il l'est d'autant plus que les constantes de temps c_i sont élevées, car les probabilités d'accéder au cache augmentent et ainsi le nombre de requêtes transférées vers le réseau de capteurs diminue. Il est à noter que dans l'hypothèse d'un rythme de requête élevé, il est particulièrement intéressant de mettre en cache. En raison des collisions de paquets et des situations de trafic élevé, l'énergie consommée lors de ces situations est considérable. Ainsi en plus des requêtes qui sont économisées on gagne également sur les coûts liés aux collisions de paquets.

4 Adaptation du cache pour optimiser la satisfaction utilisateur

Afin d'avoir une utilisation plus efficace des ressources, nous allons élaborer une technique utilisant à la fois les volontés des utilisateurs mais aussi les objectifs de l'application à rester en fonctionnement pour une certaine durée de vie. L'optimisation a deux buts, le premier est de fournir une configuration de durée de cache admissible pour les exigences de satisfaction de l'utilisateur. Le second but, réciproque du premier, est d'estimer quelle peut- être la satisfaction de l'utilisateur pour une durée de vie du réseau imposée.

4.1 Modélisation de la satisfaction utilisateur

Nous introduisons les quantités $c_{i_{\min}}$ et $c_{i_{\max}}$ représentant les bornes admissibles d'une valeur c_i. Ces valeurs exogènes au problème dépendent fortement de l'application en question. Une valeur haute de c_i sera utilisée pour des nœuds ayant peu d'énergie. Par contre, une valeur c_i petite sera pour des informations devant être aussi fraiches que possible. Pour un intervalle de temps $[c_{i_{\min}}, c_{i_{\max}}]$, nous pouvons voir que les meilleures durées de vie possible sont obtenues quand $c_i = c_{i_{\max}}$ mais cela se fait généralement au détriment de la satisfaction de l'utilisateur.

Il est nécessaire d'introduire également une métrique de satisfaction de l'utilisateur. Pour notre modélisation nous allons considérer que plus une information donnée à l'utilisateur est fraiche, plus ce dernier est satisfait. Cependant, à rythme de requête égal, une durée de vie dans le cache courte signifie plus de requêtes pour le réseau contraint et donc un coût énergétique plus important.

$$\gamma_i = \frac{c_{i_{\max}} - c_i}{c_{i_{\max}} - c_{i_{\min}}} * 100 \tag{3}$$

En particulier, une satisfaction de 0% quand $c_i = c_{i_{\max}}$ et une satisfaction de 100% quand $c_i = c_{i_{\min}}$.

4.2 Analyse énergétique et modélisation du réseau

Les nœuds utilisent ContikiMAC comme protocole d'accès à la couche liaison afin d'éviter les collisions de transmissions et contrôler l'accès au médium. L'utilisation de l'interface radio étant le principal consommateur de la batterie dans nos simulations, nous avons analysé ce protocole afin d'estimer la durée de vie du réseau. Il y a 4 états possibles pour un nœud utilisant ContikiMAC : (i) transmission, (ii) réception, (iii) mode sommeil et (iv) écoute du canal avec les consommations énergétiques suivantes : Ω_{T_i}, Ω_{R_i}, Ω_{S_i}, et Ω_{CL_i} (dimension :[W]), respectivement.

$$\Omega_{\mathrm{tot}_i} = \Omega_{\mathrm{S}_i} + \Omega_{\mathrm{CL}_i} + \Omega_{\mathrm{T}_i} + \Omega_{\mathrm{R}_i} \tag{4}$$

où : Ω_{S_i} est l'énergie consommée pendant la période de sommeil du i-ème nœud ; Ω_{CL_i} est l'énergie demandée pour écouter le canal sur une période fixée ; Ω_{R_i} est l'énergie utilisée pour la réception d'un paquet ; Ω_{T_i} est l'énergie utilisée pour transmettre un paquet. Nous ne pouvons pas agir efficacement sur les termes Ω_{S_i} et Ω_{CL_i} car ils dépendent fortement du protocole MAC utilisé. C'est sur Ω_{T_i} et Ω_{R_i} que nous allons agir via notre cache. Une modélisation plus précise peut-être trouvée dans [LML13].

Il est également nécessaire de prendre en compte la topologie du réseau puisque les messages relayés par un nœud affectent tous les nœuds dont il dépend pour la transition de ses paquets. Le modèle devenant complexe dans le cas général, nous avons donc développé une calcul approché de la consommation énergétique qui intègre la topologie du réseau.

4.3 Résultats

Afin de trouver un compromis acceptable entre durée de vie du réseau et satisfaction utilisateur, nous utilisons une méthode d'optimisation multiobjectif NSGA II (Non-dominated Sorting Genetic Algorithm II) [DPAM02] combinée à une modélisation exposée en détail dans [LML13]. Comme présenté dans la Figure 3, nous obtenons pour chaque durée de vie admissible la meilleure satisfaction utilisateur γ_i possible [LML13]. Nous obtenons ainsi un front de Pareto permettant de choisir parmi les solutions optimales celle qui est adaptée à notre application en fonction des informations provenant des différentes couches logiques.

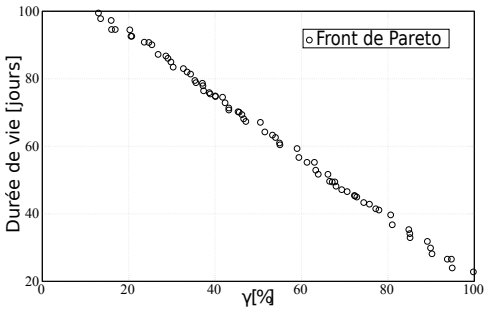

FIGURE 3: Front des optimums de Pareto

La raison pour laquelle nous obtenons un front de Pareto en forme de droite affine provient de nos hypothèses de départ. En effet, le rythme d'arrivée des requêtes est supposé uniforme, et tous les nœuds sont identiques. Dans ces conditions, nous obtenons une loi linéaire sur les défauts de cache et la consommation du réseau qui en dépend. Cependant, dans le cas où la couche MAC pourrait être dynamiquement configurée en fonction des informations disponibles, la relation serait beaucoup plus complexe et la modélisation linéaire impossible.

5 Conclusions et perspectives

Cet article [†] a présenté une stratégie pour améliorer la qualité de service d'un réseau de capteurs en adaptant dynamiquement les paramètres de la mémoire cache en fonction d'informations en provenance de différentes couches logiques du système. En premier lieu, nous avons vu qu'un cache standard aidait à économiser significativement de l'énergie puisqu'il empêche les communications redondantes triviales. Puis nous avons introduit un modèle d'optimisation exploitant les différents critères des utilisateurs ainsi que les informations extraites du cache. Ce qui nous a permis de configurer de manière optimale la gestion de la durée de vie des informations au sein du cache.

Une ouverture possible de ces travaux serait d'étudier les mécanismes de gestion des routes en temps réel au sein du réseau en tenant compte des informations obtenues dans le cache. La passerelle pourrait décider de temporiser certaine requêtes afin de synchroniser les requêtes avec l'état des nœuds dans le réseau. En outre, il serait également possible en connaissant la topologie du réseau de connaitre les modifications de topologie à faire afin d'obtenir un réseau plus résilient. Une autre ouverture serait d'avoir une méthode de calcul exacte pour le calcul d'optimisation. Enfin, une exploitation de nos simulations sur une plateforme de plus grande envergure, telle que Senslab, permettrait d'obtenir des informations supplémentaires sur la consommation énergétique physique des capteurs.

Références

[DGV04] A. Dunkels, B. Gronvall, and T. Voigt. Contiki-a lightweight and flexible operating system for tiny networked sensors. pages 455–462, 2004.

[DPAM02] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm : NSGA-II. *IEEE Trans. on Evolutionary Computation*, 6(2) :182–197, April 2002.

[LML13] Rémy Leone, Paolo Medagliani, and Jérémie Leguay. Optimizing QoS in Wireless Sensors Networks using a Caching Platform. In *Sensornets 2013*, page 56, Barcelone, Espagne, February 2013.

[She12] Z. Shelby. Constrained Application Protocol (CoAP). Internet-Draft draft-ietf-core-coap-13, Internet Engineering Task Force, March 2012. Work in progress.

†. Ce travail a été financé par la Communauté Européenne à travers le projet CALIPSO (Bourse 288879).